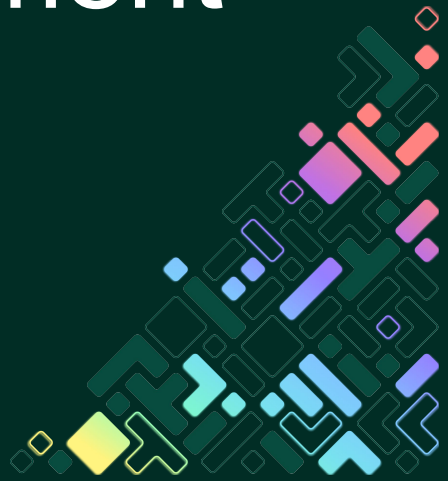
EMBEDDED
LINUX
CONFERENCE

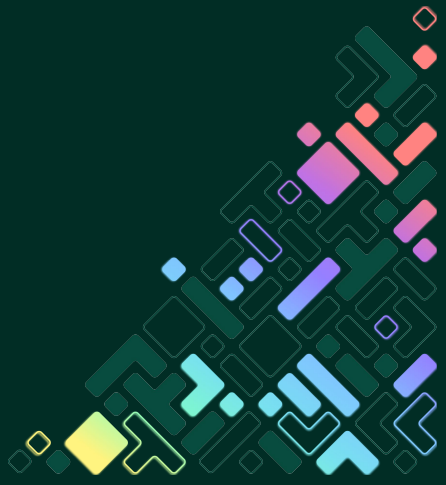
Overcoming Challenges in Embedded OSS Development



THE LINUX FOUNDATION

#EmbeddedOSSummit @handle

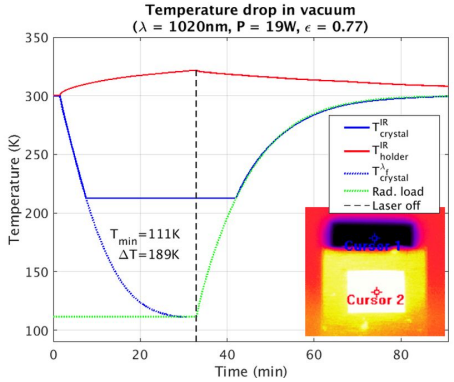
Lessons from OSS in space

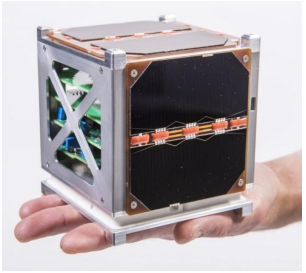



SPACE
IS HARD

Lessons from OSS in space

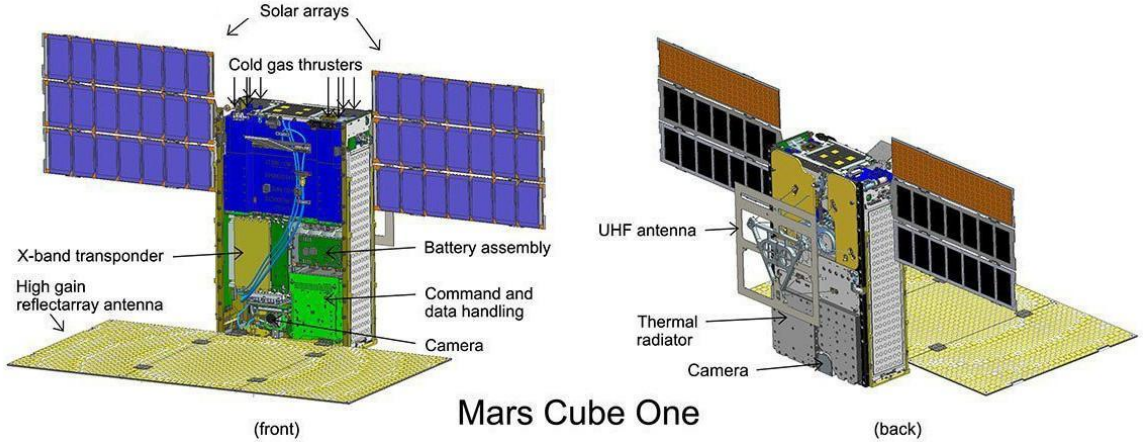
- So many constraints
  - Hardware issues: Extreme Temperature variations, Radiation, Pressure (Vacuum), Vibration
  - Limits on: Power, Physical size, Weight (every gram counts)
  - Requirements: Performance, Fault tolerance, Realtime, Power management
- Extremely high cost per mission
  - Low units: often 1 unit
  - High cost of design, testing, hardware, launch, operations
  - Failure is "not an option" (but a high percentage of cubesats fail)
    - This is why craft often last longer than intended
      - Over-engineered, for robustness





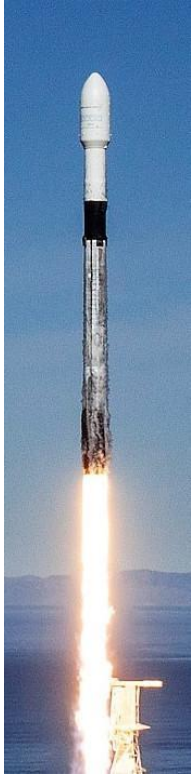
Space missions use LOTS of custom hardware

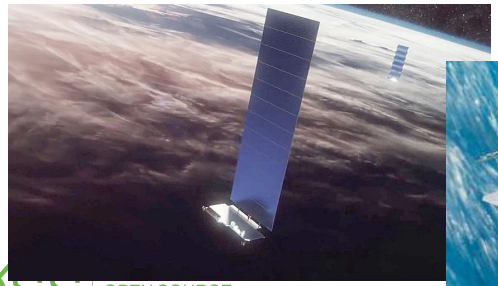
- Focus of mission is on specialized science and commercial tasks ('the payload')
  - Almost every payload has unique, bespoke hardware
- Even base systems use novel hardware
  - Thrusters, batteries, stabilizers, power units, sensors, etc.
  - Every mission seems to want to try something new

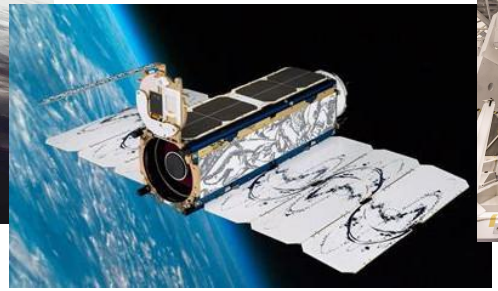


Exceptions: COTS hardware reuse

- SpaceX rockets
  - triple-redundant pairs of COTS x86 processors
- Starlink and Planet satellite constellations
  - x86 processors, not rad-hardened
- Mars Ingenuity helicopter and Perseverance rover and backshell
  - Used some off-the-shelf parts:
    - Qualcomm processor, COTS sensors, USB busses and hubs

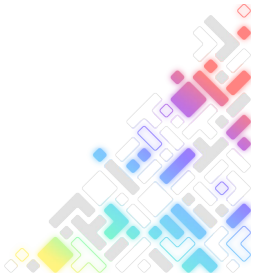






Why am I talking about OSS in space?

- Space sector is kind of "embedded on steroids"
  - Made me hearken back to my youth...
- Emblematic of issues that show up in embedded systems
  - Constraints (power, performance, real-time)
  - Custom-purpose devices and software
  - Hard to find people to collaborate with (for some parts of the stack)



Open Source means collaboration

What defines Open Source?

- Open Source is defined by the ability to use, but also *contribute* to an open code base
- Not the same thing as "source available"

- Two effects are key to Open Source

Many Minds Effect

Problem Solver Effect

The "Many Minds" Effect

Many Minds Effect

- Variety of experiences and skills results in better ideas

- Open Source tries to achieve a meritocracy, where the best ideas win
- Light bulb analogy:
  - Ideas for a project are like light bulbs...

Many Minds Effect (cont.)

- Small community = small number of ideas

Many Minds Effect (cont.)

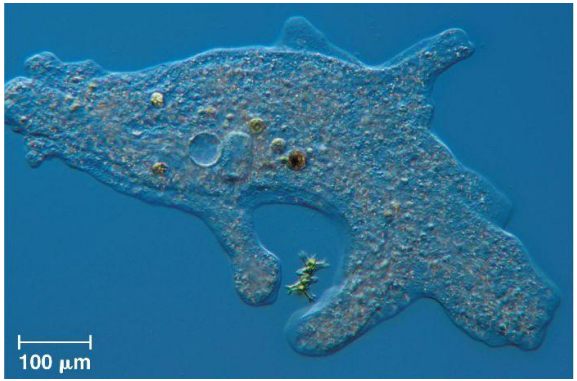
- Small community = small number of ideas
- Bigger community = more ideas
  - Better probability that a really good idea will emerge
  - Is a mathematical principle

Many Minds Effect (for bugs)

- "Given enough eyeballs, all bugs are shallow"
- I prefer to break it into parts:
  - With the right eyeballs, you can fix any bug
  - With enough eyeballs, the right eyeballs will be available

- *This was important in finding and mitigating the 'XZ backdoor'*

Problem Solver effect



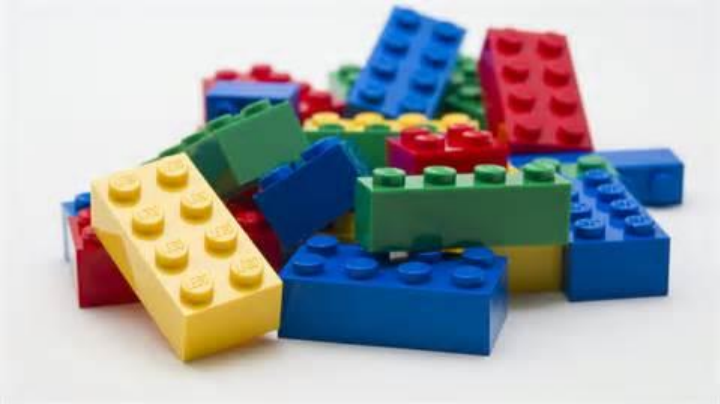
- Problems are solved as they are encountered

- Software must come “in contact” with the problem space to advance
- Most software is written to solve a specific problem
  - It does not grow outside of it’s original niche
- Openness of OSS allows it to encounter other problem spaces
  - It can adapt and grow in ways different from the original use case

- The OSS virtuous circle: The more problems a piece of software solves, the more users it attracts, and the bigger its community gets

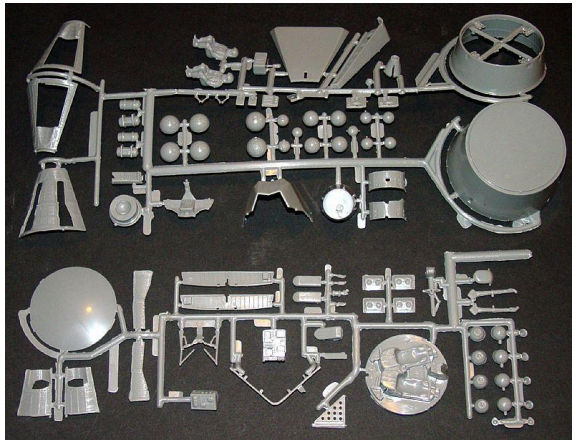
Generalization vs. Specialization

Generalization vs. specialization



Legos

- Modular
- Interchangeable
- Reusable

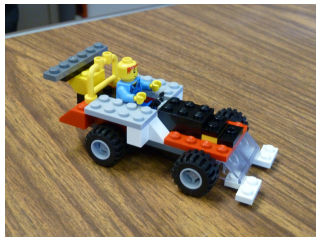


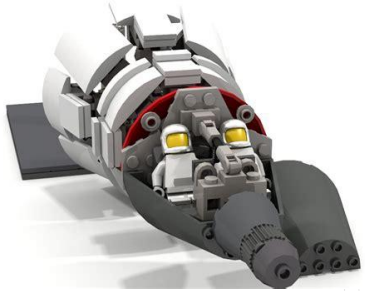
Parts for a model space capsule

- Custom
- Specific
- Fit-for-purpose

Generalization vs. specialization (cont).

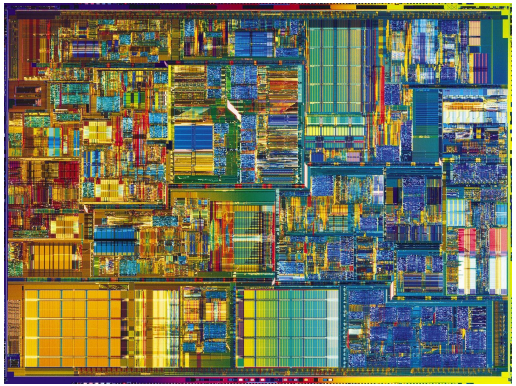
- Spaceship pieces are really good for making a spaceship
- With Lego pieces, you can make also make a spaceship
  - But you can also make a boat, or a car, or a house
- Admittedly, a spaceship made of spaceship pieces will be better
- But the Lego pieces are more general and versatile





Open Source prefer “legos”

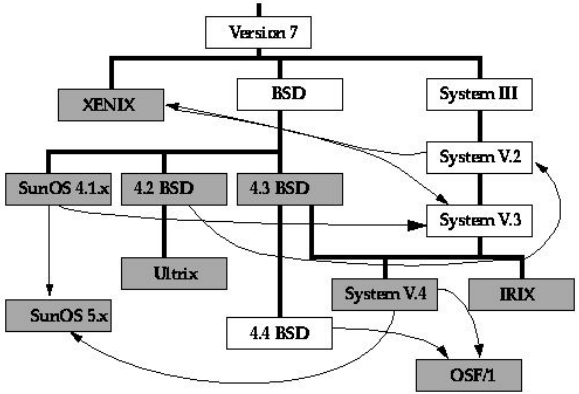
The same is true of modern hardware



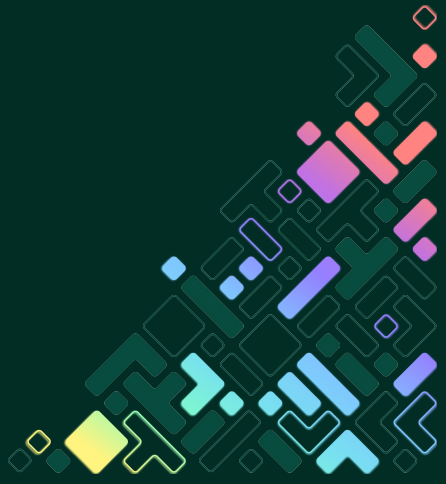
- A modern processor has "too much" stuff on it
- Why? – because the processors have been generalized so they can support a wide variety of tasks
  - Commoditization of mobile phone hardware has made processors and hardware features for embedded very cheap
  - There is now a processor that can run Linux, that costs 15 cents
- Your embedded app is unlikely to use every IP block on a modern processor
  - That's like the rough edges and extra "nubs" on a lego model

Fragmentation – the bane of OSS

- Fragmentation is when communities become divided
- Unix famously fragmented into different camps
  - Different APIs meant that it was difficult to create an application ecosystem
  - Unix Fragmentation -> autotools -> XV backdoor
- Divided community = reduced open source effects



Why Linux, then?

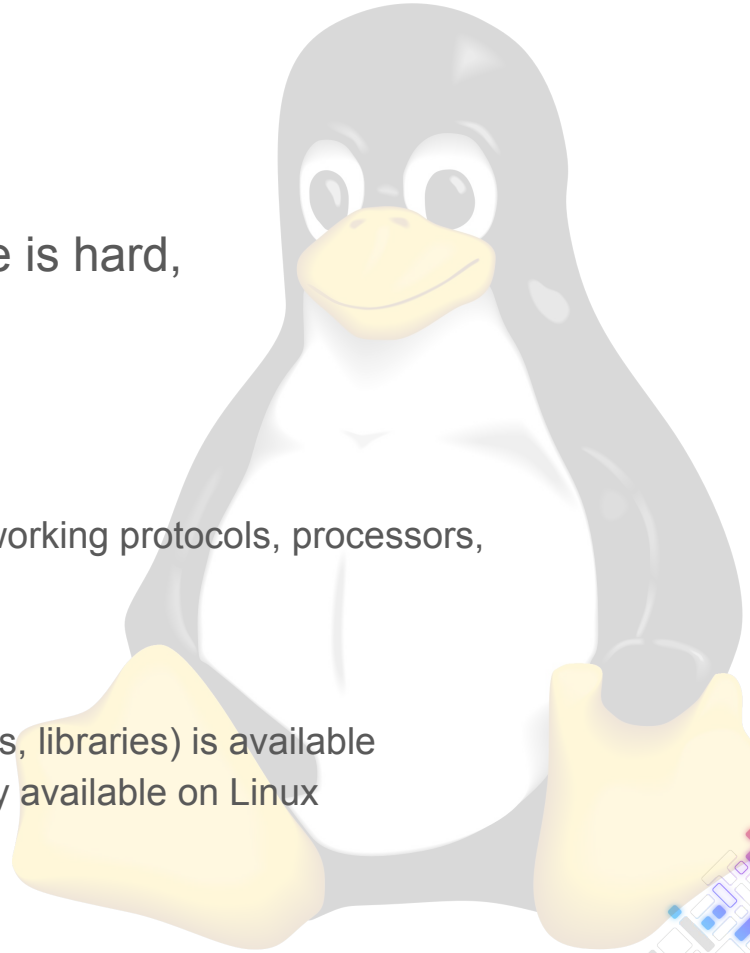


Why Linux, then?

- If working with OSS in the embedded space is hard,

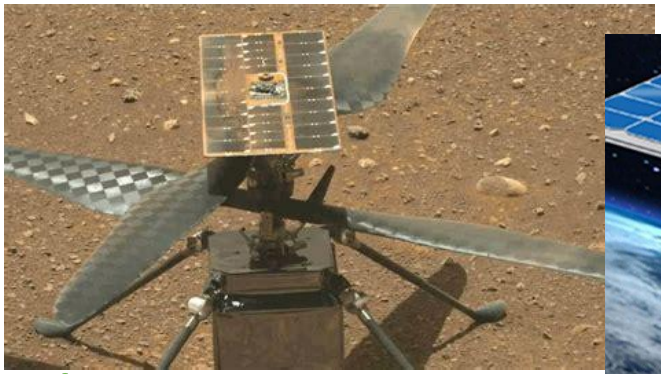
  why do it?

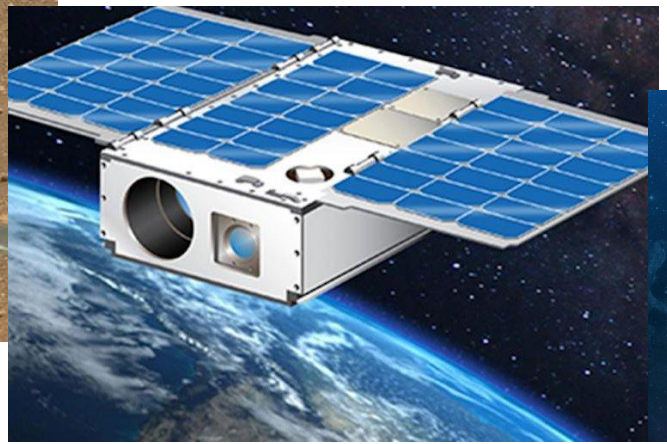
- Linux Benefits:
  - What you need may already exist
    - e.g. driver support for sensors, busses, networking protocols, processors, filesystems, storage systems, etc.
  - Share development cost with others
  - Easy to find developers experienced with Linux
  - A vast array of open source software (tools, drivers, libraries) is available
    - Some technologies (e.g. AI and ML) are only available on Linux

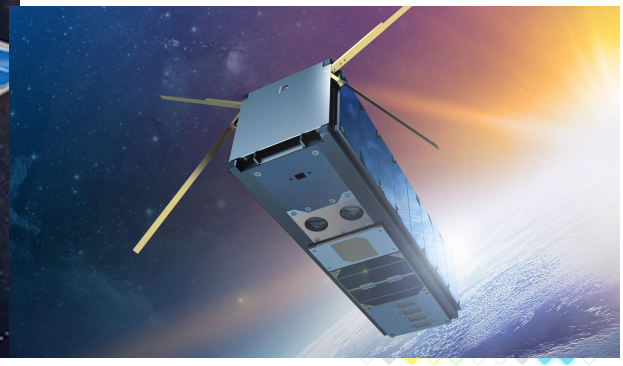


Anecdotes from the space sector

- Some space missions used shell scripts and Linux distro features to extend capabilities or resolve issues
- Ingenuity used compression to solve a problem, when not in the original plan
  - Because, hey, gzip is lying around in the distribution anyway
- Asteria and Aalto cubesats



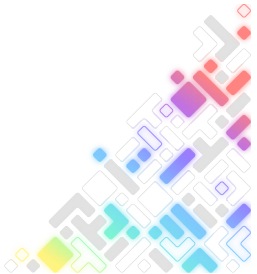




How to overcome the challenges of OSS in embedded

How to overcome the challenges of OSS in embedded

- Increase the community
- Improve generalization
- Don't specialize where you don't need to
- Find allies



Increase the community

- Contributors come from users
  - You have to have users in order to increase the pool of contributors
- Join the community
  - Be a part of the visible user base
  - Join an existing community rather than start your own
- Enhance the community
  - Invite others
  - Do something to make the community more valuable

Do Something in the Community

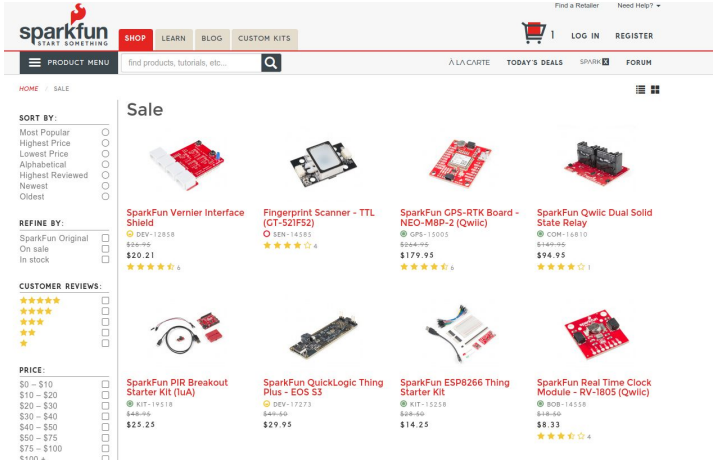
- Contributions can be in many forms
  - Usage reports
  - Bug reports
  - Documentation
  - Infrastructure management
  - Testing
  - Reviewing
  - Marketing and advocacy
  - Contribute code

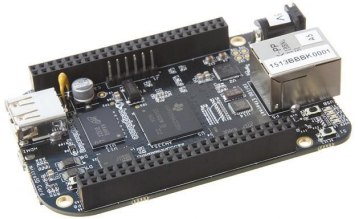
Improve generalization

- Extend existing mechanisms rather than add new ones
- Make sure your contributions handle other people's use cases

Don't specialize where you don't need to

- Use the same hardware that others are using
- Use the same sub-systems and software technologies as others
- Don't over-reduce
  - Ship with more than the absolute minimum you need





Find allies

- Find people who care about your issues, and work with them
- Sometimes, it's not who you expect:
  - Small system size
    - For hardware-constrained (e.g. IOT) devices
    - Also: secure systems want reduced attack surface
  - Quick boot
    - Automotive Linux, Consumer electronics
    - Also: Desktop users want fast boot
    - Also: Cloud servers – for quick CPU resource spinup
  - Low power usage
    - Mobile phone developers, IOT developers
    - Also: Data Center Linux developers

Conclusion

You can overcome the challenges of doing embedded OSS development

- Make the world a better place!
- Make your own job easier

EMBEDDED LINUX CONFERENCE